Bab 3

Bahasa SQL

Di bab ini akan diceritakan sekilas mengenai sejarah SQL, lalu dibahas elemen-elemen dasar bahasa SQL.

3.1 Sejarah SQL

Seperti telah diceritakan di Bab 2, di awal dan pertengahan tahun 1970-an, vendor database berlomba-lomba mengimplementasikan database relasional yang berbasiskan model relasional seperti yang dijabarkan dalam paper E. F. Codd. IBM sendiri kala itu tengah mengembangkan produk relasionalnya yang bernama System/R. Sebagai bahasa antarmuka terhadap System/R ini, IBM menciptakan bahasa query yang bernama SEQUEL. Ciri-ciri bahasa ini adalah mirip bahasa Inggris, deklaratif, dan high-level. Barangkali hal ini karena terinspirasi kesuksesan bahasa COBOL. Perlu diketahui, program COBOL memang amat *verbose* dan lebih mirip kalimat-kalimat bahasa Inggris ketimbang bahasa komputer.

Ternyata System/R tidaklah terlalu sukses di pasaran, sehingga IBM mengakhirinya tahun 1979 (IBM belakangan memiliki produk database relasional lain yaitu DB/2). Namun SEQUEL ternyata dilirik oleh banyak vendor lain, seperti Oracle dan Ingres, yang ikut-ikutan menggunakan varian dialek SEQUEL ini di dalam produk mereka. Belakangan nama SEQUEL diubah menjadi SQL oleh IBM dikarenakan adanya persengketaan merek dengan perusahaan lain. Namun untuk menghormati atau mengenang sejarahnya, hingga kini SQL dieja “sequel” (baca: si-kwel).

Di awal 1980-an, sudah ada beberapa produk database yang menggunakan SQL. Badan standar Amerika, ANSI, akhirnya mengadopsi SQL menjadi standar tahun 1986. Setahun berikutnya ISO pun mengangkat SQL menjadi standar. Standar pertama ini sering disebut SQL-86 atau SQL-87.

Standar berikutnya adalah SQL-89, yang merupakan revisi minor dari SQL-87. Dokumen standar SQL-89 terdiri dari sekitar 100 halaman. Saat itu, standar SQL belum memasukkan schema, full outer join, dan cascade update/delete untuk foreign key constraint.

SQL-92 merupakan generasi kedua standar SQL dan sering disebut SQL2. Tebal dokumen standar melonjak menjadi sekitar 600 halaman. SQL-92 menambahkan banyak hal, antara lain: information schema, berbagai tipe join, union di view, tipe-tipe data tanggal, domain, ALTER TABLE, CASE, dan lain sebagainya. Rata-rata produk database yang ada sekarang memiliki kompliansi terhadap SQL-92 pada tingkat basic.

SQL:1999 merupakan generasi ketiga standar SQL dan dijuluki SQL3. Pada tahap ini, SQL sudah memasukkan fitur-fitur yang dianggap “kontroversial” dan tidak bersifat murni relasional lagi. Di antaranya yaitu fitur OO seperti table inheritance, tipe data komposit (array, row), dan tipe data referensi (“pointer”). Selain itu fitur-fitur lain yaitu recursive query, regex, trigger, tipe data boolean (sebelumnya hanya

dikenal tipe data BIT untuk menyimpan 0/1), dan savepoint. Banyak produk database yang telah mengimplementasi berbagai fitur SQL:1999, meskipun tidak ada yang mengimplementasi penuh semuanya.

SQL:2003 merupakan standar SQL terbaru. Salah satu fasilitas yang diperkenalkan yaitu yang berkaitan dengan XML.

3.2 SQL dan Model Relasional

SQL dibuat pertama kali oleh IBM sebagai "bahasanya database relasional". Karena itu rata-rata konsep di SQL secara langsung merupakan terjemahan langsung akan apa yang ada pada model relasional. Misalnya, tabel dan view di SQL adalah relasi. Tupel di model relasional merupakan row (baris). Deretan atribut dalam relasi merupakan kolom. Dikenal juga domain, referential integrity, key, dan lain sebagainya.

Meskipun demikian, seiring perkembangannya SQL kini telah menyimpang dari model relasional ideal. Kini SQL memiliki fitur-fitur seperti pointer (reference type) yang tidak dikenal di model relasional. Standar SQL:1999 dan SQL:2003 juga memasukkan unsur-unsur model hierarkis (seperti XML atau recursive query untuk data tree). Sebaliknya, sampai kini pun SQL masih belum bisa 100% memenuhi ketiga belas aturan yang telah diutarakan pada tahun 1970 di paper Codd.

Oleh para idealis relasional (para relasionalis?) SQL seringkali disebut "pseudo-relational" atau bahasa query relasional yang pincang atau tak sempurna. Sudah beberapa kali juga ada upaya untuk membuat bahasa query baru (atau modifikasi SQL) yang bisa lebih memenuhi kaidah model relasional. Namun karena SQL telah begitu familiar bagi banyak orang, dan didukung oleh begitu banyak produk database besar dan kecil, hingga kini belum ada yang sanggup menggantikan atau bahkan mendekati popularitas SQL. Bahkan beberapa produk database relasional yang sebelumnya mendukung lebih dari satu bahasa query selain SQL, akhirnya memberhentikan dukungan tersebut dan kini hanya mendukung SQL. Contoh kasusnya adalah Interbase dengan GDML-nya, Ingres dengan Quel-nya, dan PostgreSQL dengan Postquel-nya. Semuanya akhirnya punah rata-rata dengan alasan bahwa bahasa-bahasa lain tersebut tidaklah terlalu sering dipakai penggunanya dan daripada harus memelihara dua buah bahasa, hanya SQL yang dipertahankan.

Satu hal yang perlu diingat oleh para pemula adalah: SQL tidaklah identik dengan model atau database relasional itu sendiri. (Bahkan model relasional dan database relasional adalah dua hal yang juga berbeda). SQL merupakan salah satu cara

berinteraksi dengan database relasional. Analogikan ini dengan bahasa shell Bash di Unix/Linux yang dipakai untuk berinteraksi dengan kernel Linux itu sendiri, atau web browser yang digunakan untuk berinteraksi dengan Web. Penyebab ambiguitas ini sendiri adalah karena memang praktis nyaris semua database relasional yang ada sekarang interface utamanya hanyalah SQL.

3.3 Mengapa SQL?

Model relasional bukanlah satu-satunya cara untuk menyimpan data. SQL juga bukan satu-satunya cara untuk berinteraksi dengan database. Tapi kenapa mayoritas orang memilih kedua kombinasi ini untuk menaruh dan mengolah data mereka?

Pada akhirnya jawabannya adalah, bahwa secara keseluruhan manfaat yang diperoleh paling besar. Mari bandingkan database relasional SQL dengan jenis-jenis database lain.

Menyimpan data di file teks (*flat file*) memiliki keuntungan yaitu data mudah dibaca oleh manusia dan mudah diedit secara manual menggunakan editor teks. Untuk menggali informasi dari file ini di Unix/Linux secara sederhana bisa dipakai utilitas teks, misalnya cat, head, tail, dan grep. Tapi di luar kasus-kasus sederhana, biasanya diperlukan program untuk memparse tiap baris atau paragraf file teks ke dalam struktur data variabel sebelum melakukan pengolahan terhadap variabel-variabel tersebut.

Menyimpan data dalam bentuk XML, YAML, dan lain sebagainya pada dasarnya keuntungan dan kerugiannya sama dengan format file teks. Bisa dibaca manusia, tapi belum terdapat bahasa query standar yang disediakan produk database.

Menyimpan data dalam worksheet seperti Excel amat nyaman dari segi kemudahan diedit, visualisasi/graphing, pemformatan, dan lain sebagainya. Untuk query sederhana pun tersedia tool seperti AutoFilter, Find, dan Pivot Table. Namun untuk query yang lebih kompleks, diperlukan penulisan dalam bahasa program. Menyimpan data di spreadsheet hanya cocok untuk data berukuran kecil (mis: sampai dengan 65 ribu baris saja). Dan tidak ada dukungan seperti transaksi, akses multiuser, mode klien-server, atau aspek manajemen database lainnya.

Menyimpan data di database sederhana seperti BerkeleyDB atau CDB memiliki keuntungan yaitu kemudahan dan kecepatan. Namun lagi-lagi, untuk melakukan query atau manipulasi diperlukan penulisan program dalam salah satu bahasa pemrograman.

Menyimpan data dalam bentuk dump isi memori variabel (seperti `serialize()` di PHP, pickle/shelve di Python, atau Storable di Perl) memiliki keuntungan yaitu tidak perlu melakukan parsing dan penyimpanan/pemuatan amat sederhana. Tapi, biasanya formatnya spesifik untuk sebuah bahasa pemrograman, hanya layak untuk yang ukuran kecil (karena pada umumnya harus dimuat secara keseluruhan di memori) dan, lagi-lagi, untuk melakukan query yang kompleks kita harus menulis program.

Bisa dilihat polanya di sini bahwa keuntungan utama menggunakan database relasional SQL adalah tidak perlunya kita menulis program berpuluh atau ratus (atau ribu) baris untuk melakukan query atau reporting. Memang benar kita juga tetap harus menulis query dalam SQL, tapi karena bahasa SQL bersifat deklaratif dan high-level, maka untuk melakukan sebuah query kompleks jumlah baris dan waktu yang dibutuhkan untuk menulis query SQL jauh lebih pendek dibandingkan harus menulis program. Dan karena bahasa SQL itu merupakan lingua franca berbagai macam produk database relasional, maka database relasional menjadi cukup portabel antarproduk dan antarbahasa pemrograman. Sejak pertengahan 1990-an, saya telah belajar lebih dari 7-8 bahasa pemrograman, menggunakan beberapa sistem operasi dan banyak distro Linux, namun pengetahuan akan bahasa SQL terus terpakai hingga sekarang. Data yang ada dulu di database relasional SQL dengan mudah masih bisa kita akses dan manipulasi dengan bahasa yang sama hari ini.

3.4 Seperti Apa Bahasa SQL?

Bahasa SQL diciptakan tahun 1970-an, jadi tidaklah terlalu mirip seperti bahasa-bahasa yang berkembang tahun 1990-an seperti PHP, Perl, atau Python. Targetnya pun tidak hanya untuk programer, sehingga bahasa ini cenderung lebih mendekati bahasa Inggris dan tidak terlalu banyak mengandung simbol atau operator yang sulit diingat. Kalau ingin dibandingkan, mungkin lebih dekat ke COBOL dan FORTRAN ketimbang Python atau PHP.

Bahasa SQL adalah bahasa yang bersifat deklaratif, tidak prosedural; walaupun ada varian bahasa SQL untuk menulis stored procedure yang bersifat prosedural. Karena itu, SQL tidaklah secara eksplisit mendukung deklarasi variabel, statement untuk looping, statement untuk percabangan (IF … THEN …), dan lain sebagainya. Sebuah query SQL yang kompleks pun dinyatakan dalam sebuah statement saja.

Bahasa SQL tidak bersifat case-sensitive. Anda dapat menulis kata-kata kunci SQL dalam huruf besar atau huruf kecil, meskipun biasanya kebiasaan umumnya adalah

menulis dalam huruf besar semua. Nama-nama tabel dan kolom juga umumnya tidak bersifat case-sensitive (meskipun kadang case-sensitive, misalnya, nama tabel di MySQL menjadi nama file, dan nama file di Unix sensitif sementara di Windows case-insensitive).

Bahasa SQL juga bersifat high level. SQL tidak mengurusi lokasi fisik seperti offset byte sebuah record atau nama file untuk sebuah tabel, melainkan mengizinkan kita memanipulasi database, tabel, baris, dan kolom tanpa mengetahui di mana sebetulnya objek-objek itu berada di disk atau dalam format apa disimpannya. MySQL dan PostgreSQL misalnya, menyimpan database masing-masnig dalam format yang berbeda (bahkan di MySQL, antara satu engine tabel dengan engine tabel lain cara penyimpanannya dapat berbeda drastis), namun Anda tetap dapat menggunakan perintah SQL yang sama untuk memanipulasinya.

3.5 Elemen Sintaks SQL

Statement

Bahasa SQL terdiri dari statement atau kalimat atau perintah. Statement adalah unit dasar dalam bahasa SQL. Antara satu statement dengan yang lain dipisahkan dengan tanda titik koma. Sebuah statement dapat ditulis dalam beberapa baris—dipotong-potong dengan newline—tanpa mengubah artinya. Ini seringkali berguna jika Anda menulis query yang amat panjang dan untuk memudahkan pembacaan dipecah-pecah ke dalam banyak baris dan disisipi baris-baris kosong.

Catatan: Kadang-kadang titik koma tidak dibutuhkan, misalnya jika Anda mengirimkan statement SQL lewat bahasa pemrograman PHP menggunakan `mysql_query()` di PHP, maka tidak dibutuhkan titik koma karena hanya satu statement yang dikirimkan.

Statement SQL umumnya digolongkan ke dalam DML, DDL, dan DCL.

Kelompok pertama, DML (data manipulation language), adalah perintah/kalimat-kalimat SQL yang berguna untuk mengquery database, memasukkan data (baris), serta mengupdate atau menghapus data. Kelompok ini adalah perintah-perintah SQL yang akan paling sering Anda gunakan. Untuk mengquery database, digunakan perintah SELECT. Untuk memasukkan data, ada perintah INSERT dan MERGE. Untuk mengupdate data, UPDATE. Untuk menghapus data, DELETE. Untuk mengosongkan sebuah tabel, ada pula perintah bernama TRUNCATE.

Kelompok kedua, DDL (data description/definition language), berurusan dengan pembuatan, pengubahan, atau penghapusan tabel itu sendiri. Untuk membuat

tabel, digunakan perintah CREATE TABLE. Untuk mengubah dan menghapus tabel masing-masing perintahnya ALTER TABLE dan CREATE TABLE.

Kelompok ketiga, DCL (data control language), berhubungan dengan pengaturan akses ke data. Yang termasuk dalam kelompok ini adalah GRANT (untuk memberi user akses terhadap sejumlah perintah) dan REVOKE (untuk mencabut akses yang sebelumnya di-GRANT).

Selain tiga kelompok besar ini, ada juga perintah-perintah lain seperti COMMIT dan ROLLBACK yang berhubungan dengan transaksi, DECLARE/FETCH/MOVE/CLOSE yang berhubungan dengan cursor, dan lain sebagainya.

SELECT merupakan statement SQL yang akan paling sering Anda gunakan (katakanlah, 90%). Sisanya sebagian besar adalah INSERT, UPDATE, dan DELETE. Baru sisanya lagi perintah-perintah lain. Karena itu di buku ini fokus pembahasan adalah mengenai query (SELECT).

Komentar

Komentar di SQL dapat disisipkan di awal, tengah, atau akhir statement. Komentar diawali dengan tanda `--` (dua buah tanda minus) dan berlaku hingga akhir baris. Contoh:

```
-- contoh komentar di awal
SELECT
  id,
  nama,
  AVG(nilai) -- contoh komentar di tengah
FROM daftarnilai
WHERE time BETWEEN '2004-08-01' AND '2004-08-31';
-- contoh komentar di akhir
```

Selain komentar satu baris menggunakan `--`, standar SQL juga memperbolehkan komentar ala C yang bisa berlaku hingga banyak baris:

```
/* ini komentar baris pertama
   masih komentar
   masih komentar
   akhir komentar */
```

Spesifik di MySQL: Sebetulnya standar SQL mengizinkan komentar gaya C ini bersarang (komentar di dalam komentar) sbb:

```
/*
 * ini komentar.
 *
 *  /* dan ini komentar dalam komentar. */
 *
 */
```

Namun saat ini hal tersebut tidak diperbolehkan di MySQL.

Keyword dan Identifier

Sama seperti mayoritas bahasa pemrograman lainnya, SQL pun mengenal kata-kata kunci (keyword), yaitu kata-kata yang memiliki arti khusus dan tidak dapat dipakai sebagai nama-nama kolom dan tabel (identifier), kecuali jika identifier tersebut dikutip.

Keyword SQL antara lain SELECT, CREATE, TABLE, UPDATE, dan lain sebagainya. Daftar lengkap keyword SQL yang digunakan MySQL ada di manual MySQL (jumlahnya sekitar 200-an). Jadi kata-kata seperti SELECT atau CREATE tidak dapat dijadikan sebagai nama tabel atau kolom, kecuali jika Anda mengutipnya dengan notasi kutip terbalik (‘SELECT‘, ‘CREATE‘). Di standar SQL, tanda kutip identifier adalah kutip ganda (“SELECT”, “CREATE”) namun di MySQL defaultnya kutip ganda juga digunakan sebagai pengutip nilai string di samping kutip tunggal. Kalau Anda ingin dapat menggunakan juga kutip ganda sebagai pengutip identifier sama seperti di database lain, Anda dapat menjalankan MySQL dalam mode ANSI (menjalankan `mysqld` dengan `opsi --ansi`).

Keyword dan identifier harus diawali dengan huruf atau garis bawah dan diikuti nol atau lebih huruf, angka, atau garis bawah. Namun, jika disebutkan dengan dikutip, Anda bisa menggunakan simbol dan spasi. Catatan: di MySQL Anda tetap dilarang membuat identifier yang diakhiri dengan spasi (trailing blanks), meskipun identifier tersebut dikutip.

Identifier bersifat case-insensitive kecuali jika dikutip.
Berikut ini contoh identifier:

```
satu
Satu        -- sama dengan satu, karena case-insensitive
‘satu‘      -- sama dengan satu, bedanya hanya dikutip
‘Satu‘      -- tidak sama dengan satu, karena case-sensitive
            --   (dikutip). Sama dengan Satu.
satu dua    -- tidak diperbolehkan
```

```
'satu dua'  -- diperbolehkan, karena dikutip
' satu'     -- diperbolehkan, karena dikutip
'satu '     -- tidak diperbolehkan di MySQL, meskipun dikutip
```

Nilai Literal

Nilai literal angka ditulis seperti biasa, dengan notasi seperti 123, -123.45, atau 1.23e3.

Nilai literal string ditulis dengan kutip tunggal, misalnya: `'Andi'` atau `'Andri Firmansyah'`. Di MySQL Anda juga dapat menggunakan kutip ganda, namun ingatlah bahwa menurut standar dan di rata-rata database lain kutip ganda digunakan bukan untuk mengutip string tapi mengutip identifier.

Di dalam literal string, Anda dapat menggunakan escape sequences sebagaimana yang sering dijumpai dalam C atau PHP, seperti `\n` (yang berarti newline), `\t` (yang berarti tab), `\\` (yang berarti garis miring terbalik literal), `\'` (yang berarti kutip tunggal literal), `\0` (NUL, karakter ASCII berkode 0), dan lain sebagainya. Daftar lengkap kode escape ada di manual MySQL.

Catatan untuk MySQL: Selain kedua jenis literal di atas, MySQL juga memperbolehkan kita menyebutkan literal string dalam deretan heksadesimal (mis: `0x63` berarti huruf `'c'`). Jika ingin membuat literal angka dari kode heksadesimal, gunakan ekspresi seperti `0x63+0`, yang berarti 99.

Operator

SQL mengenal operator-operator matematika seperti `+`, `-`, `*`, `/`. Catatan: untuk pangkat, di MySQL Anda menggunakan fungsi `POW()`. Misalnya, `POW(2, 8)` sama dengan 2 pangkat 8 yaitu 256. Sebetulnya di standar SQL digunakan simbol `^` (caret), tapi di MySQL simbol ini digunakan untuk operasi bit XOR.

Operasi bit di MySQL menggunakan operator & (AND), | (OR), ^ (XOR), dan ! (NOT). Selain operasi bit, padanannya adalah operator logika && (AND), || (OR), `xor` (XOR), dan `not` (NOT). Catatan: di standar SQL simbol || digunakan untuk menyambung string, sementara di MySQL digunakan fungsi `CONCAT()`. Standar SQL pun tidak mendukung operasi XOR.

Selain operator ini, terdapat pula operator CASE, BETWEEN, IN, OVERLAPS, IS TRUE, IS FALSE, IS UNKNOWN, IS NULL, IS NOT NULL. Di buku ini nanti akan ada contoh-contoh resep SQL yang menggunakan operator tersebut.

3.6 Tipe Data di SQL

SQL menyediakan beberapa jenis tipe data dasar: boolean, bilangan bulat, bilangan pecahan desimal, teks, data biner, tanggal/waktu. Sebetulnya standar SQL:1999 menyediakan beberapa lagi tipe data seperti tipe data komposit (array dan row) serta tipe data ref/pointer. Standar SQL:2003 pun menambahkan tipe data seperti MULTISET (yang mirip array) dan XML. Namun banyak database, termasuk MySQL, yang tidak atau belum mendukungnya, jadi kita tidak membahas tipe-tipe data ini. Untuk sebagian besar aplikasi, jenis-jenis data dasar sudah mencukupi dan akan membuat data Anda cukup portabel (tidak terlalu bergantung pada sebuah produk database tertentu).

Nyaris semua produk database menyediakan tipe data ekstra atau jenis data yang agak berbeda dengan yang didefinisikan oleh standar. PostgreSQL misalnya, memiliki tipe data alamat IP. MySQL memiliki SET dan ENUM, dan kelakuan tipe data tanggalnya menyimpang dari standar.

Tipe-tipe data ekstra atau varian-variannya inilah yang kadang-kadang menjadi ganjalan utama portabilitas, mempersulit kita pindah dari satu database ke database lainnya dikarenakan tidak adanya padanan tipe data yang spesifik atau yang kelakuannya sama persis pada database semula di produk database tujuan. Contohlah misalnya Microsoft SQL Server yang menyediakan tipe data GUID. Di database lain, kolom-kolom bertipe GUID ini harus dikonversi, entah ke dalam tipe teks, blob, atau gabungan 2 kolom BIGINT. Atau MySQL yang memiliki tipe data TIMESTAMP yang berkelakuan khusus (autoupdate) atau yang tipe data DATE-nya memperbolehkan tanggal-tanggal yang tidak valid (mis: 2004-02-30 alias 30 Feb 2004).

Karena itu, jika Anda memperhatikan portabilitas database, maka kekanglah diri Anda untuk sebisa mungkin menggunakan tipe data dasar saja yang dikenal standar SQL. Dan tidak memanfaatkan kelakuan-kelakuan spesifik tipe data yang ada di produk database yang Anda pakai.

Boolean

`BOOL[EAN]`

Boolean menyimpan nilai benar atau salah. Boolean baru diperkenalkan (distandarkan) di standar SQL:1999, meskipun banyak database yang sudah menyediakannya sejak bertahun-tahun sebelumnya. Untuk database yang tidak menyediakannya, biasanya digunakan CHAR(1) (teks 1 huruf) atau SMALLINT

(integer berkapasitas 16bit) yang diberi constraint agar hanya dapat menyimpan nilai T/F (true/false) atau 0/1.

Catatan untuk MySQL: BOOLEAN baru dikenal di MySQL 4.1. Dan di MySQL tipe data ini saat ini sebetulnya hanyalah alias untuk TINYINT(1) dan Anda dapat memasukkan nilai di luar 0/1 (mis: 2, 100, dan lain sebagainya). Nilai 0 artinya salah dan semua nilai bukan 0 artinya benar. Menurut manual, MySQL berencana membuat tipe data boolean "sejati" di kemudian hari.

Bilangan bulat

INT[EGER]
SMALLINT
BIGINT

Data bilangan bulat dapat disimpan dalam 3 pilihan tipe data INT, SMALLINT, atau BIGINT. Perbedaannya adalah pada kapasitas angka yang dapat disimpan. INT berkapasitas 32bit (k.l. -2 milyar s.d. +2 milyar), SMALLINT berkapasitas 16bit (-32.768 s.d. +32.767), dan BIGINT berkapasitas 64bit (-9 juta triliun s.d. +9 juta triliun). SMALLINT cocok untuk menyimpan data seperti umur, angka tahun, dan jumlah-jumlah kecil lainnya yang tidak diharapkan melebihi puluhan ribu. INT bisa digunakan untuk sebagian besar keperluan umum. BIGINT cocok untuk menyimpan angka yang besar seperti jumlah uang (jika disimpan dalam satuan sen), nomor kartu kredit, dan lain sebagainya. Percaya atau tidak, BIGINT sudah didukung oleh database sejak bertahun-tahun lalu, tapi baru di standar SQL:2003 ditambahkan.

Spesifik di MySQL: MySQL menyediakan modifier UNSIGNED yang dapat disebutkan setelah tipe data: INT UNSIGNED, SMALLINT UNSIGNED, BIGINT UNSIGNED. Tipe data versi unsigned ini tidak dapat menyimpan bilangan negatif, tapi kapasitasnya dari 0 hingga 2x nilai terbesar versi signednya. Contohnya INT UNSIGNED dapat menyimpan dari 0 hingga 65.535.

Selain itu MySQL menyediakan dua tipe data bilangan bulat lainnya: MEDIUMINT dan TINYINT. TINYINT berkapasitas hanya 8bit, sehingga hanya dapat menyimpan -128 s.d. 127 (atau 0 s.d. 255 untuk versi unsignednya). MEDIUMINT berkapasitas 24bit, sehingga dapat menyimpan -8 juta s.d. 8 juta (atau 0 s.d. 16 juta untuk versi unsignednya). Rata-rata produk database lain tidak mengenal UNSIGNED maupun MEDIUMINT/TINYINT, jadi hindari kecuali jika ingin database menjadi spesifik MySQL. Hanya gunakan INT/SMALLINT/BIGINT demi portabilitas.

Bilangan pecahan desimal

```
REAL | FLOAT
DOUBLE[PRECISION]
DECIMAL[(x,y)]
```

Bilangan pecahan desimal dapat disimpan dalam kolom bertipe REAL (atau FLOAT), DOUBLE PRECISION, dan DECIMAL. FLOAT berkapasitas 32bit, sering juga disebut memiliki ketelitian "tunggal" (6 angka di belakang koma dsimal). DOUBLE PRECISION membutuhkan 64bit, disebut memiliki ketelitian "ganda" (15 angka di belakang koma).

Kedua tipe bilangan pecahan FLOAT dan DOUBLE PRECISION sebetulnya disimpan dalam basis 2 dan bersifat tidak eksak karena tidak bisa sempurna menyimpan sebagian bilangan berbasis 10. Jika digunakan dalam operasi aritmetika, maka dapat timbul kesalahan pembulatan (round-off error). Kedua tipe data ini tidak cocok digunakan untuk menyimpan jumlah yang harus bersifat eksak, seperti jumlah uang (selisih 1 sen maka siap-siaplah dituntut pelanggan!).

Karena itu, untuk menyimpan bilangan pecahan desimal yang eksak berbasis 10, disediakan tipe bilangan pecahan ketiga: DECIMAL. Tipe data DECIMAL benar-benar disimpan dalam basis 10 sehingga bersifat eksak (sebagian database di balik layar menyimpan bilangan DECIMAL sebagai BIGINT plus informasi lokasi titik desimal, sehingga semua operasi aritmetika dilakukan dengan operasi bilangan bulat untuk mencegah terjadinya round-off error).

Tipe data DECIMAL memberi kita pilihan: DECIMAL(x, y) di mana x adalah presisi (jumlah digit total maksimum) dan y adalah scale (jumlah digit setelah koma desimal). Contoh, DECIMAL(5,2) berarti kita dapat menyimpan bilangan seperti 123,45 atau -999,99 atau 1,01; tapi tidak dapat menyimpan 1234,5 (alias 1234,50; presisi=6) atau 123,001 (akan dibulatkan menjadi 123,00).

Jika *x* dan *y* tidak disebutkan, maka MySQL akan menganggap *x*=10 dan *y*=0. Banyak produk database memiliki nama alias untuk DECIMAL, antara lain NUMERIC, DEC, FIXED.

Catatan untuk MySQL: Di MySQL, REAL adalah sinonim untuk DOUBLE (ketelitian ganda) dan bukannya FLOAT (ketelitian tunggal). Ini berbeda dari konvensi umum, karena itu MySQL menyediakan mode operasi REAL_AS_FLOAT yang jika dipilih akan membalikkan arti REAL menjadi sinonim untuk FLOAT dan bukan DOUBLE.

Untuk amannya, gunakan “FLOAT” dan “DOUBLE” agar tidak perlu mengalami kerancuan ini.

Teks

```
CHAR[ACTER][(n)]
VARCHAR | CHARACTER VARYING [(n)]
```

Data teks dapat disimpan dalam kolom bertipe CHAR (dapat juga ditulis CHARACTER) atau VARCHAR (dapat juga ditulis CHARACTER VARYING). CHAR untuk menyimpan teks yang panjangnya sama semua untuk semua baris (misalnya: kode negara ISO alpha-2, yang selalu terdiri dari 2 huruf; atau nomor ISBN, yang selalu terdiri dari 10 karakter). VARCHAR untuk menyimpan teks yang panjangnya dapat berbeda-beda tiap baris (misalnya: nama, alamat, dan lain sebagainya). Perbedaan keduanya sebetulnya hanya dalam urusan efisiensi penyimpanan saja. Jika kita mendefinisikan sebuah kolom sebagai CHAR(100) maka semua baris akan mengalokasikan 100 karakter untuk kolom tersebut, tak peduli meskipun sebuah kolom hanya menyimpan teks yang pendek. Sementara untuk VARCHAR, database mengalokasikan ruang disk yang berbeda-beda sesuai panjang isi kolom—karena itulah namanya VARCHAR atau CHARACTER VARYING. VARCHAR(100) berarti panjang *maksimum* 100, bukan semua dialokasikan 100 karakter.

Di rata-rata produk database, CHAR dan VARCHAR biasanya dimaksudkan dan dibatasi untuk menyimpan teks yang tidak terlalu panjang (hanya sampai ratusan karakter saja). Untuk teks ribuan karakter ke atas, biasanya produk database menyediakan tipe data CLOB atau variannya. MySQL misalnya, menyediakan tipe data TEXT (64KB), MEDIUMTEXT (16MB), dan LONGTEXT (4GB). Postgres menyediakan tipe data bernama TEXT yang dapat menampung sekitar 1GB. Sementara Firebird sudah menyediakan kapasitas yang cukup panjang bagi VARCHAR, sekitar 32KB, dan tidak menyediakan tipe data CLOB.

Data biner

```
BLOB
```

Data biner (gambar, multimedia, deretan byte mentah) dapat disimpan dalam tipe kolom BLOB (BLOB adalah singkatan dari binary large object). Kolom BLOB juga tentu saja bisa berfungsi sebagai CLOB. Postgres kini menggunakan tipe data bernama BYTEA, yang kurang lebih sama dengan BLOB. Beberapa produk database, termasuk MySQL, menyediakan beberapa varian BLOB: TINYBLOB (255 byte), BLOB (64KB), MEDIUMBLOB (16MB), dan LONGBLOB (4GB).

Data tanggal dan waktu

```
DATE
TIME [(p)] [WITHOUT TIME ZONE]
TIME [(p)] WITH TIME ZONE
TIMESTAMP [(p)] [WITHOUT TIME ZONE]
TIMESTAMP [(p)] WITH TIME ZONE
INTERVAL [(p)]
```

Data tanggal dan/atau waktu disimpan dalam kolom bertipe DATE, TIME, TIMESTAMP. DATE untuk menyimpan tanggal (mis: 2004-12-25 untuk 25 Des 2004). TIME untuk menyimpan waktu (mis: 10:01:02 untuk jam 10 pagi lewat 1 menit 1 detik). TIMESTAMP untuk menyimpan tanggal dan waktu. TIME dan TIMESTAMP dapat memiliki TIME ZONE.
SQL juga menyediakan tipe data INTERVAL untuk menyimpan periode/jangka waktu (mis: “2 day” atau “1 hour 10 minute”). Ini mempermudah kita dalam melakukan aritmetika tanggal dan waktu..

Tipe data TIMESTAMP dan INTERVAL dapat diberi opsi ketelitian (*p*) yaitu untuk menyatakan berapa jumlah angka ketelitian untuk detik. *p*=3 berarti kita mencatat ketelitian hingga 0.001 detik (milidetik).

Sayang sekali, dukungan MySQL untuk tipe data tanggal/waktu amat terbatas dan menyimpang jauh dari yang didefinisikan oleh standar SQL. Tidak ada dukungan untuk zona waktu, tidak ada dukungan presisi di bawah 1 detik, tidak ada tipe data INTERVAL, aritmetika tanggal/waktu canggung, diperbolehkannya tanggal tidak sah dimasukkan, TIMESTAMP memiliki arti khusus yang spesifik, dan lain sebagainya.

Spesifik di MySQL: Di MySQL, tipe data untuk menyimpan tanggal dan waktu disebut DATETIME. Ketelitiannya hanya hingga 1 detik. Di MySQL juga ada tipe data bernama TIMESTAMP, namun artinya sedikit berbeda. TIMESTAMP sama dengan DATETIME untuk menyimpan tanggal dan waktu, namun memiliki kelakuan “ajaib” yaitu akan otomatis mengupdate dirinya sendiri setiap kali sebuah baris tabel di-UPDATE. Kelakuan ini mirip dengan atribut mtime (tanggal terakhir modifikasi file) di filesystem. (Automatic updating ini sebetulnya dapat dimatikan jika MySQL berjalan dalam mode operasi MAXDB. Pada mode ini, TIMESTAMP menjadi identik dengan DATETIME. Namun karena TIMESTAMP telah ada sejak bertahun-tahun di MySQL, rata-rata aplikasi mengharapkan automatic updating.)

Selain itu, MySQL menyediakan tipe data YEAR(2) dan YEAR(4) untuk menyimpan tahun saja. Hindari tipe data ini kecuali jika ingin database Anda spesifik untuk MySQL.

Data geometri

MySQL sejak versi 4.1 menambahkan tipe-tipe data geometri (spatial data) seperti titik, garis, poligon, dan lain sebagainya. Spatial data juga didukung oleh database seperti Oracle dan PostgreSQL. Memang tidak semua aplikasi membutuhkan tipe data geometri; yang membutuhkan antara lain aplikasi pemetaan, sains, dan lain sebagainya.

Sebetulnya database relasional bisa saja mengimplementasi tipe-tipe data geometri ini melalui komposisi koordinat (misalnya, titik diimplementasi dengan dua kolom DOUBLE x dan y). Namun tipe data khusus geometri ini memiliki keuntungan dapat diindeks dengan spatial index (R-tree) sehingga memungkinkan query-query seperti "cari semua kota yang berjarak tidak lebih dari 300 km dari kota Bandung" dengan cepat.

3.7 NULL di SQL

Menurut model yang diajukan oleh E. F. Codd, database relasional haruslah menyediakan cara untuk dapat merepresentasikan informasi yang hilang/tidak diketahui (missing information) atau informasi yang tidak berlaku (not applicable, N/A). Contoh informasi yang hilang/tidak diketahui misalnya jika kita menerima form isian pengunjung website dan si pengunjung website tidak mengisi data jenis kelaminnya. Sementara contoh informasi yang tidak berlaku misalnya jenis kelamin saat diterapkan untuk perusahaan, bukan orang.

Namun di SQL, hanya ada satu cara untuk merepresentasikan kedua hal tersebut yaitu menggunakan nilai khusus yang disebut NULL. Pemahaman akan apa NULL itu dan bagaimana kelakuannya merupakan salah satu kunci pemahaman SQL. NULL bisa disetarakan seperti `unset` pada PHP atau `None` pada Python atau `undef` pada Perl. Tapi kelakuannya tidak persis sama.

Dua sifat "aneh" NULL adalah: pertama, NULL tidaklah benar maupun salah. NULL bukanlah nol atau string kosong. NULL tidak sama dengan nilai manapun juga, termasuk tidak sama dengan NULL itu sendiri! Ini dapat dipahami jika kita mengingat bahwa NULL adalah perwakilan untuk "sesuatu yang tidak diketahui". Jadi kita tidak mengetahui apakah sesuatu itu benar atau salah atau bernilai seperti apa. Dan jika kita memiliki dua buah kotak hitam yang tidak diketahui isinya, maka kita tidak bisa berkesimpulan bahwa kedua kotak tersebut isinya sama. Sehingga dengan analogi ini, NULL = NULL bernilai salah.

Kedua, operasi dengan NULL akan membuat nilai NULL "menular". Contohnya, jika kita melakukan operasi 2 + NULL, maka hasilnya adalah NULL. NULL / 0 pun menghasilkan NULL. Demikian pula jika kita menggunakan fungsi dan salah satu argumennya bernilai NULL, maka keseluruhan hasil fungsi menjadi NULL. Kembali, jika kita mengingat bahwa NULL adalah perwakilan untuk "missing information", jika ada operasi yang membutuhkan sejumlah masukan dan salah satu masukan tidak diketahui nilainya, maka otomatis operasi tersebut tidak dapat menghasilkan kepastian. Sehingga hasil akhirnya NULL.

3.8 SQL di MySQL

MySQL adalah sebuah database relasional gratis dan open source (GPL) yang mula-mula tersedia di Unix/Linux namun kini tersedia juga di sistem operasi lain seperti Windows. MySQL mulai popular sejak pertengahan 1990-an saat Web dan aplikasi web mulai popular. Kala itu, selain MySQL, tidak ada alternatif database lain yang cepat, stabil, dan memiliki fitur-fitur yang cukup untuk bisa dijadikan database pendukung aplikasi web. Hingga kini MySQL terus bertahan sebagai database open source yang paling popular mengalahkan PostgreSQL, Interbase/Firebird, dan lain sebagainya. MySQL mudah diinstal, mudah dipakai, dan dapat dikonek dari berbagai bahasa pemrograman. Singkat kata, amat aksesibel bagi rata-rata pengguna yang ingin mulai berdatabase ria.

Kekuatan utama MySQL adalah pada kecepatannya, terutama untuk kecepatan koneksi (overhead koneksi yang rendah) dan kecepatannya untuk query-query yang sederhana. Hal ini membuatnya cocok dipakai sebagai backend untuk aplikasi web termasuk yang berbasis CGI.

MySQL juga menyediakan fitur-fitur yang cukup membantu dalam pembuatan aplikasi web seperti klausa LIMIT dalam SELECT, full text index, dan recovery database yang mudah. Ini wajar karena salah satu pemakaian utama MySQL adalah untuk aplikasi web.

Dalam hal kompliansi dengan standar SQL, perlu diakui masih banyak yang harus dilakukan pengembang MySQL dalam menyusul saingan-saingannya seperti PostgreSQL, Firebird, atau Ingres. Ada banyak fitur seperti view, trigger, stored procedure, atau check constraint yang sudah merupakan makanan sehari-hari di database-database lain namun belum juga muncul di MySQL (stored procedure sudah mulai muncul di MySQL 5.0, yang saat buku ini ditulis masih berstatus alpha; sementara view dan trigger belum ada tanda-tandanya kapan akan mulai diimplementasi).

Belum lagi kelakuan MySQL banyak yang tidak sesuai standar. MySQL biasanya cenderung terlalu "pengampun" dalam menerima input data. Kalau database lain menolak data masukan dengan pesan error, maka MySQL dengan santai akan menerima data tersebut—setelah diubah sedikit atau dipotong diam-diam. Beberapa kelakuan ini akan dibahas dalam buku agar para pengguna MySQL tidak terkejut saat berpindah ke database lain.

Secara keseluruhan, MySQL merupakan database yang bisa digunakan oleh pemula atau mereka yang ingin database yang bersifat lightweight dan cepat. Untuk yang membutuhkan fitur-fitur database SQL yang lebih banyak, barangkali ada pilihan lain seperti PostgreSQL yang bisa dipertimbangkan, atau menunggu versi MySQL berikutnya yang menjanjikan akan mengimplementasi semua fitur standar SQL.

Buku ini tetap mengambil MySQL sebagai patokan sebab itulah yang dipakai oleh sebagian besar pembaca. Namun dengan tidak melupakan standar SQL atau fasilitas-fasilitas yang mungkin ada di database lain.